根据应用描述检查应用行为

Alessandra Gorla Ilaria Tavecchia[*] 弗洛里安Gross Andreas Zeller
萨尔大学萨尔布吕肯，
德国
{gorla，tavecchia，fgross，zeller}@cs.uni-saarland.de

摘要

我们如何知道一个程序做它声称做的事情？ 在根据描述主题对Android应用程序进行聚类后，我们根据各自的API使用情况确定每个群集中的异常值。 发送消息的“天气”应用程序因此变得异常；同样，通常不会期望“消息传递”应用访问当前位置。 应用于一套22,500多种Android应用程序，我们的CHABADA原型确定了一些异常情况；此外，它标记了56％的新型恶意软件，而不需要任何已知的恶意软件模式。

类别和主题描述

D.4.6 [安全和保护]：有创软件

一般条款

安全

关键词

Android，恶意软件检测，描述分析，聚类

1. 介绍

检查程序是否执行它声称的做法是开发人员长期存在的问题。 不幸的是，它现在也已经成为计算机用户的问题。 无论何时我们安装新应用程序，我们都会冒这个应用程序为“恶意软件”的风险，即违反其用户的利益。

到目前为止，研究和行业都致力于通过针对预定义的恶意行为模式检查静态代码和动态行为来检测恶意软件。 但是，这对新的攻击没有帮助，因为很难预先定义某些程序行为是有益的还是恶意的。 问题在于任何使行为变得有利或者恶意的规范都取决于当前的情况。 举例来说，在移动世界中，在一个应用程序中被视为恶意行为可能是另一个应用程序的功能：

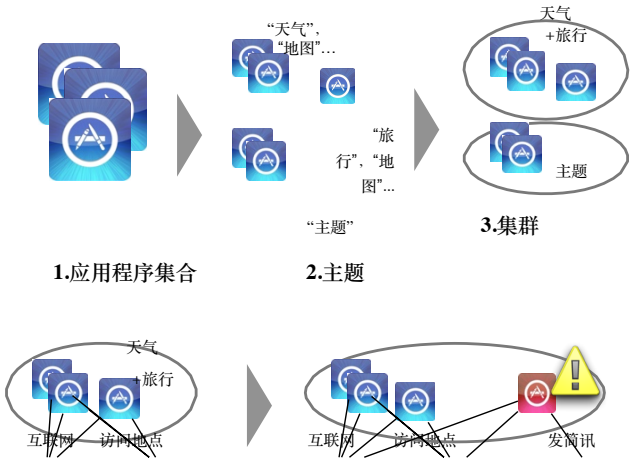


图1：检测具有未公开行为的应用程序。 从一批“好”应用程序（1）开始，我们确定其描述主题（2）以形成相关应用程序的集群（3）。 对于每个集群，我们确定使用的有感知的API（4），然后可以识别使用对该集群不常见的API的异常值（5）。

- 发送短信到付费号码筹款的应用程序可疑吗？ 也许，但在Android上，这是解锁游戏功能的合法付款方式。
- 追踪您当前位置的应用程序是恶意的？ 如果它是导航应用程序，路径跟踪器或地图应用程序，则不是。
- 将所有联系人发送给某个服务器的应用程序都是恶意的？ 这是WhatsApp在初始化时所做的事情，它是世界上最流行的移动消息应用程序之一。

因此，问题不在于应用程序的行为是否与特定模式匹配; 这是程序是否像广告一样行事。 在上述所有示例中，用户都会被告知并在出现任何可疑行为前被要求授权。 这是一种可疑或彻头彻尾的恶意行为。

在本文中，我们试图检查实施的应用程序行为与广告的应用程序行为。 我们的域名是Android应用程序，因其市场份额以及攻击和欺诈历史而被选中。 作为应用广告行为的代理，我们使用Google Play商店中的自然语言描述。 作为其实施行为的代理，我们使用从内部使用的一组Android应用程序编程接口（API）

[*] Ilaria Tavecchia现在在SWIFT，比利时布鲁塞尔。

图2：“导航和旅行”群集的应用程序描述中的单词。 单词越大，它出现的应用程序描述就越多。

图3：“个性化”应用程序中的单词

应用程序二进制。 关键的想法是将描述和API用法联系起来以检测异常情况：“这个'天气'应用程序访问消息传递API，这对于这个类别来说是不寻常的。”

具体来说，我们的CHABADA方法[1] 采取五步，如图所示图1 并在后面的文章中详述：

1. CHABADA以从Google Play商店下载的22,500多款“好”Android应用程序开始。

2. 在应用程序描述中使用潜在Dirichlet分配（LDA），CHABADA确定每个应用程序的主题（“主题”，“地图”，“天气”，“下载”）。

3. 然后CHABADA将相关主题的应用程序分组。 例如，与“导航”和“旅行”相关的应用在他们的描述中共享几个主题，因此它们形成一个集群。

   图2 在“导航和旅行”群集中显示用于描述应用程序的词语云。 他们的文字云非常不同， 例如，比较“个性化”群集的单词云 （数字 3).

4. 在每个集群中，CHABADA标识每个应用程序静态访问的API。 我们只考虑受用户权限管辖的敏感API。 例如，与互联网访问相关的API受到“INTERNET”权限的控制。

   图4 显示由“导航和旅行”应用程序使用的敏感API，按照管理权限分组。 显然，这些应用的“正常”行为是访问当前位置并访问互联网（通常是地图）

图4：在“导航和旅行”群集中使用的按照关联权限分组的API。 标准行为是访问该位置（“ACCESS-FINE-LOCATION”）并访问互联网（“ACCESS-NETWORK-STATE”，“INTERNET”）。

图5：“个性化”群集的API。 与之相反 图4， 这些应用程序经常访问SD卡（“WRITE-EXTERNAL-STORAGE”），启用和禁用新功能（“CHANGE-COMPONENT-ENABLED-STATE”），但很少，
如果有的话，访问设备位置。

[1] CHABADA代表针对应用程序描述的CHecking应用程序行为。“Chabada”是爵士乐的基本三重节奏模式的法语单词。

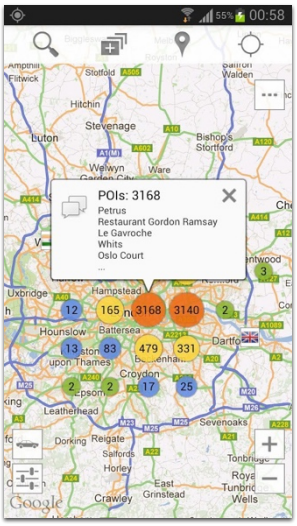
服务器）。 互联网接入也被“个性化”应用使用（数字 5）；但是，他们将访问外部存储和组件，而不是当前位置。

5. 使用无监督的One-ClassSVM异常分类，CHABADA可以确定与API使用有关的异常值。 它会为每个群集生成一个应用程序的排名列表，其中顶级应用程序的API使用情况最为异常 - 表明说明和实现之间可能存在不匹配。 同样，尚未知的应用程序将首先被分配给其描述所隐含的集群，然后被归类为正常或异常。

   在“导航和旅行”集群中，任何改变电话或其组件的配置的API的使用都是不寻常的；但是，这对于“个性化”应用很常见，而这些应用很少访问当前位置。 在这两个群集中，任何会读取或编写联系人，日历条目，图片，文本消息，浏览器历史记录，网络设置等的应用程序都会立即被标记为异常值，因此需要进一步审查。

通过标记每个群集内异常的API使用情况，CHABADA的设立是为了检测广告和实施行为之间的任何类型的不匹配。 这在实践中如何工作？ 图6 显示伦敦餐厅和酒吧+[2]，可从Google Play商店获得。 其描述清楚地将其放入“导航和旅行”集群中。 但是，除了期望的API调用访问当前位置和Internet外，它还使用三个API调用来检索device-getAccountsByType（），getDeviceId（）和getLine1Number（）中的用户帐户列表，全部由“ GET-ACCOUNTS“权限。 这些呼叫检索诸如设备标识符和移动电话号码等敏感信息，使得伦敦餐厅成为“导航和旅行”中的异常点。 事实上，伦敦餐厅将这些信息（包括诸如当前位置的附加信息）秘密地发送到其广告服务，这在描述中根本没有提及。

[2] https://play.google.com/store/apps/details?id=com.alarisstudio.maps.restaurants.london



寻找一家餐厅，一家酒吧，一家酒吧或只是为了在伦敦玩得开心？ 不再搜索！ 这个应用程序有你需要的所有信息：
* 你可以搜索你想要的每一种食物 法国，英国，中国，印度等。
* 如果你坐车，骑自行车或步行，你可以使用它
* 您可以查看地图上的所有目标
* 你可以搜索目标
* 您可以查看您附近的目标
* 您可以查看方向（可视路线，距离和持续时间）
* 您可以在街景视图中使用它
* 您可以在导航中使用它

关键词：伦敦，餐厅，酒吧，酒吧，食品，早餐，午餐，晚餐，餐，吃，晚餐，街景，导航

INTERNET
GET_ACCOUNTS
ACCESS_WIFI_STATE
ACCESS_NETWORK_STATE
ACCESS_FINE_LOCATION
READ_PHONE_STATE
VIBRATE

图6：应用程序伦敦餐厅酒吧和酒吧+，以及访问的完整描述和API组

这是恶意软件吗？ 有可能。 这是意外的行为吗？ 当然。[3] 如果伦敦餐厅明确了它的功能，那么它就会落入“广告”集群，而不再是一个离群值。

在我们的研究中，我们发现了更多虚假广告，普通欺诈，伪装以及其他可疑行为的例子。 作为一种副作用，我们的方法作为恶意软件检测器也很有效：在良性应用程序上训练每个集群的SVM分类器，CHABADA标记了56％的已知恶意软件，而不需要任何有关恶意软件模式的培训。

本文的其余部分安排如下。 我们首先详细介绍如何按照描述主题在应 第2节。 第3节 描述了在每个群集中，我们如何检测异常值与其API使用情况。 第4节 手动和自动，定量和定性评估我们的方法。 在讨论相关工作后 （部分 5), 第6节 以结论和后果结束。

2. 集群应用程序描述

CHABADA背后的直觉很简单：就他们的描述而言，类似的应用程序也应该具有相似的行为。 为此，我们必须首先确定两个描述“相似”的含义。 我们从Android应用程序的收集方法开始 （部分 2.1）。 经过初步处理 （部分 2.2），CHABADA确定了应用程序描述的主题 （部分 2.3），然后根据常见主题对应用程序进行聚类 （第2.4节 至 部分 2.6).

1. 收集应用程序

我们的方法基于检测来自“正常”的异常，希望是良性应用。 作为这种“正常”行为的基础，我们从Google Play商店收集了大量应用程序，这是Android应用程序的中心资源。 我们的自动化收集脚本在2013年冬季和春季期间定期运行，并为Google Play商店中的30个类别中的每一个类别下载免费前150名[4] 应用程序在每个类别。 一个

[3] 安装伦敦餐厅时，用户必须明确确认其权限集合，但用户为什么会找到

像“帐户访问”异常或可疑？
4.3节 讨论可能的诱发偏见。

我们的脚本的单次完整运行因此返回了4500个应用程序; 随着我们收藏中的前150个应用转移，我们在所有类别中共获得了32,136个应用。

除了实际的应用程序（以APK文件形式发布）之外，我们还收集了商店元数据 - 例如名称，说明，发布日期，用户评分或屏幕截图。 由于CHABADA将在向公众发布之前识别异常值，因此仅使用名称和描述。

2. 使用NLP预处理描述

在对我们的描述进行主题分析之前，我们将自然语言处理（NLP）的标准技术用于过滤和词干分析。

Google Play商店中的应用说明通常包含多种语言的段落 - 例如，主要描述是英文的，而在描述结束时，开发人员使用不同的语言添加短句来简要描述应用。 为了能够对类似的描述进行聚类，我们必须选择一种语言，并且由于它的优势，我们选择了英语。 要删除所有不是英文的文本段落，我们运行了Google的紧凑语言检测器[5] 检测他们最可能的语言; 非英语段落被删除。 在多语言过滤之后，我们删除了停用词（常用词，如“the”，“is”，“at”，“which”，“on”，...），并在所有描述中应用词干。 词干是一种常见的自然语言处理技术，用于识别单词的根，并且使诸如“玩”，“玩家”和“游戏”等单词与单个通用根“plai”匹配是非常重要的。 词干可以改善后来的NLP过程的结果，因为它减少了词的数量。 我们也删除了非文字诸如数字，HTML标签，链接和电子邮件地址等项目。

作为一个例子，考虑一下伦敦餐厅的描述 in 图6; 在停止词语移除和词干之后，它显示为：

> 看起来restaur酒吧酒吧只是乐趣伦敦搜索应用程序通知需要可以搜索everi类型的食物想要法国英国chines印度等我们可以汽车bicycl步行可以查看对象地图可以搜索对象可以查看对象附近可以查看直接视觉溃败distanc durat可以我们街景可以我们navig关键词伦敦restaur酒吧酒吧食物早餐午餐晚餐吃晚饭街景navig

有了这些，我们从我们的集合中删除了这些应用程序，其描述在上述NLP预处理之后将少于10个字。 此外，我们消除了所有没有任何敏感API的应用程序（请参阅 第3节 了解详情）。 这导致了最终的22,521个应用程序，这是我们方法的基础。

3. 用LDA识别主题

为了确定正在分析的应用程序的主题集，我们使用使用潜在狄利克雷分配（*LDA*）的主题建模 [4].

LDA依靠统计模型来发现未标记文本集合中出现的主题。“话题”由经常一起出现的一组词组成。 例如，通过分析一组关于导航和旅行的应用程序描述，LDA将诸如“地图”，“交通”，“路线”和“位置”等词汇分成一个集群，以及“城市”，“景点”“游览”，并“访问”到另一个群集中。 因此，描述主要关于导航的应用程序将被分配到第一个主题，因为描述中出现的大多数单词属于第一个集群。 然而，诸如伦敦餐厅之类的应用将被分配到两个主题，因为说明中的词汇出现在两个集群中。

我们的实现将NLP预处理的输出（即，没有停用词的英文文本以及词干化后）输入到Mallet框架中 [18]. 我们可以自由选择主题的数量

[5] http://code.google.com/p/chromium-compact-language-detector/

表1：Android应用开采的主题

Id	指定名称	大多数代表性词语（源于）
1.	“个性化”	galaxi，nexu，设备，屏幕，效果，安装，customis
2.	“游戏和作弊表”	游戏，视频，页面，作弊，链接，提示，技巧
3.	“钱”	插槽，机器，钱，扑克，currenc，市场，贸易，股票，赌场硬币，金融
4.	“电视”	电视，频道，countri，生活，手表，德国，国家，bbc，newspap
5.	“音乐”	音乐，歌曲，收音机，播放，播放器，听
6.	“假期”和宗教	圣诞节，万圣节，圣诞老人，一年，假期，is-林，上帝
7.	“导航和旅行”	地图，通知，跟踪，GPS，导航，旅行
8.	“语言”	语言，单词，英语，学习，德语，translat
9.	“分享”	电子邮件，广告，支持，脸谱，分享，推特，率，建议
10.	“天气和星星”	天气，预测，定位，温度，地图，城市，光
11.	“文件和视频”	文件，下载，视频，媒体，支持，年龄，分享，观看，搜索
12.	“照片和社交”	照片，朋友，脸谱，分享，爱，微博，pictur，聊天，messag，galleri，热，送社交
13.	“汽车”轨道	汽车，比赛，速度，驱动器，车辆，自行车，
14.	“设计和艺术”设计，	生活，人，自然，形式，感觉，学习，艺术，独特，效果，现代
15.	“食物和食谱”	收件人，蛋糕，鸡肉，厨师，食物
16.	“个性化”	主题，启动器，下载，安装，图标，菜单
17.	“健康”	体重，博迪，运动，饮食，锻炼，军医
18.	“旅行”	citi，guid，地图，旅行，国旗，countri，吸引
19.	“孩子和身体”趣，绘制，	孩子，动物，颜色，女孩，巴比，pictur，乐
20.	“铃声和声音”	设计，学习 声音，铃声，闹钟，notif，音乐
21.	“游戏”3D，	游戏，plai，图形，乐趣，跳转，水平，球，得分了
22.	“搜索和浏览”	搜索，图标，删除，书签，链接，homepag，快捷方式，浏览器
23.	“战斗游戏”	故事，游戏，怪物，僵尸，战争，战斗
23	“设置和使用情况”	屏幕，设置，部件，电话，batteri
24.	“体育”	团队，足球，leagu，球员，运动，篮球
25.	“壁纸”	壁纸，生活，首页，屏幕，背景，菜单
26.	“连接”	设备，连接，网络，wifi，blootooth，in-ternet，remot，服务器
27.	“政策和广告”polici，pri-	生活，广告，首页，applovin，notif，数据，vacy，share，airpush，advertis
28.	“流行媒体”	系列，视频，电影，专辑，慕维，音乐，奖，明星，粉丝，表演，江南，顶级，比伯
29.	“拼图和纸牌游戏”	游戏，plai，水平，puzzl，球员，比分，chal-leng，卡片

由LDA确定; 我们选择了30，Google Play商店中我们的应用覆

然而，我们想要的是识别具有相似描述的应用程序组，并且我们使用K-means算法这种最常见的聚类算法之一 [16].

给定度量空间中的一组元素，并给出数字K.

K-均值为每个群集选择一个质心，
然后将数据集的每个元素与最近的元素相关联

质心，从而识别集群。 在这种情况下，要聚类的要素是通过它们对主题的亲和力来确定的应用程序。

表2 显示四个应用程序应用程序 . . ，应用程序，

归属于四个主题的概率。 应用于
这组应用程序有K = 2个集群，K-means返回一个

与应用程序$_1$和应用程序$_3$的群集，以及与应用程序$_2$和应用程序$_4$的另一个群集。

表2：四个应用程序及其属于特定主题的可能性

应用	主题$_1$	主题$_2$	主题$_3$	主题$_4$	
应用程序$_1$	0.60	0.40	—	—	
应用程序$_2$	—	0.70	0.30		
应用程序$_3$	0.50	0.30	—	0.20	
应用程序$_4$	—	—	0.40	0.60	

2.5 找到最佳数量的集群

K-means的一个挑战是估计应该创建的簇的数量。 该算法需要给定一些初始的电位质心，或者簇的数量K

鉴别。 在一系列可能的解决方案中，有几种方法来确定最佳解决方案。 因此，我们运行K-means

几次，每次用不同的K数，获得一组

我们将能够评估聚类。 K的范围涵盖了两个极端之间的解决方案：拥有少量的解决方案

集群（甚至只有2个）具有各种各样的应用程序； 或者拥有许多群集（每个应用程序甚至可能有一个群集），因此具有很强的特定性 我们将num_topics 4固定为上限，因为在我们的设置中，应用程序最多可以属于4个主题。

为了确定最佳解决方案，即最佳数量的聚类，我们使用了元素轮廓，如讨论中所述 [21]. 元素的轮廓是度量元素与其群集中其他元素的匹配程度，以及其匹配程度如何松散

到相邻集群的其他元素。 当一个元素的轮廓值接近1时，表示该元素

盖的类别数量。 此外，我们建立了LDA，这样一个应用最多只能

属于4个主题，并且只有当该主题的概率至少为5％时，才会考虑与该主题相关的应用。

表 1 显示了我们分析的22,521个描述的结果列表；“分配的名称”是我们分配给该主题的抽象概念。 我们的示例应用程序伦敦餐厅分配给以下三个主题：

- 主题6（“导航和旅行”）的概率为59.8％，
- 主题14（“食物和食谱”），概率为19.9％，以及
- 主题17（“旅行”）的概率为14.0％。

4. 用K-means对应用程序进行聚类

主题建模以一定的概率为每个主题分配一个应用程序描述。换句话说，每个应用程序的特征是每个主题的亲和度值（概率）向量。

在适当的群集中。 如果值接近1，则意味着元素位于错误的簇中。 因此，为了确定最佳解决方案，我们计算元素轮廓的平均值

对于每个使用K作为聚类数量的解决方案，我们选择其轮廓最接近1的解决方案。

6. 导致的应用程序集群

表 3 显示了我们分析的22,521个应用程序所确定的群集列表。 这32个群集中的每一个都包含应用程序，其描述包含“最重要的主题”下列出的类似主题。 上一列报告的百分比代表每个集群内特定主题的权重。

我们确定的群集与在Google Play商店等应用商店中找到的类别完全不同。 例如，集群22（“广告”）充斥着应用程序，除了以某种方式展示广告外， 这些应用程序通常会承诺或提供某些用户好处作为回报 群集16（“连接”）表示处理蓝牙，Wi-Fi等的所有应用程序；Google Play商店中没有此类别。 从成人主题到宗教的几个“壁纸”集群，仅仅代表了几个应用程序提供的功能非常少的事实。

表3：应用程序集群。 “大小”是相应群集中的应用程序数量。 “最重要的话题”列出了三个最流行的话题；最重要的（> 10％）以粗体显示。 未列出的主题少于1％。

Id	指定名称	尺寸	最重要的话题
1.	“共享”	1,453	分享（53％），设置和使用情况，
2.	“拼图和纸牌游戏”	953	导航和旅行 (78%), 益智和纸牌游戏 分享，游戏
3.	“记忆困惑”	1,069	益智和纸牌游戏（40％），游戏（12％），分享
4.	“音乐”	714	音乐（58％），分享，设置和使用情况
5.	“音乐视频”	773	流行媒体（44％），节假日
6.	“宗教壁纸”	367	和宗教（20％）分享 假期和宗教（56％），设计和艺术，壁纸
7.	“语言”	602	语言（67％），分享，设置和utils
8.	“备忘单”	785	游戏和作弊表（76％），分享，流行媒体
9.	“utils的”	1,300	设置和实用程序（62％），共享，连接
10.	“体育比赛”	1,306	游戏（63％），战斗游戏，谜题和纸牌游戏
11.	“战斗游戏”（11％），	953	战斗游戏（60％），游戏
12.	“导航和旅行”	1,273	设计和艺术导航和旅游分享，旅行 (64%),
13.	“钱”	589	金钱（57％），拼图和卡片游戏，设置和使用情况
14.	“孩子们”	1,001	孩子和身体（62％），分享，
15.	“个性化”	304	益智和纸牌游戏 个性化（71％），壁纸（15％），设置和使用情况
16.	“连接”	823	连接（63％），设置和utils，share
17.	“健康”	669	健康（63％），设计和艺术，分享
18.	“天气”	282	天气和明星（61％），
19.	“体育”	580	Tings和utils（11％），导航和旅行体育（62％），分享，流行媒体
20.	“文件和视频”	679	文件和视频（63％），分享，设置和使用情况
21.	“搜索和浏览”	363	搜索和浏览（64％），游戏，
22.	“广告”	380	益智和纸牌游戏
23.	“设计和艺术”	978	政策和广告（97％） 设计和艺术（48％），分享，游戏
24.	“汽车游戏”	449	汽车（51％），游戏，谜题和
25.	“电视直播”	500	纸牌游戏 电视（57％），分享，导航和旅行
26.	“成人照片”	828	照片和社交（59％），分享，设置和使用情况
27.	“成人壁纸”	543	壁纸（51％），分享，孩子们和身体
28.	“广告壁纸”	180	政策和广告（46％），wallpa-
29.	“铃声和声音”		pers，设置和utils 铃声和声音（68％），共享，设置和使用情况

改进。 我们在此简要列出最重要的，以便将来的研究人员避免我们遇到的一些问题。

主题的用法。 有人可能会怀疑是否真的有必要根据主题进行聚类，而不是直接对明确的描述进行聚类。 原因是K-means以及任何其他聚类算法，

volved。 因此，将描述抽象为主题对获得更好的聚类结果至关重要。

集群的使用。 对于应用程序只有一个主要话题没有产生更好的结果，因为几个应用程序可能同时包含多个主题。 这也排除了给定Google Play商店类别作为群集的用法

战略。 尽管有人可能会争辩说，聚类不会产生不同的结果，而不仅仅是聚集主要的结果

主题（主题和聚类的数量几乎相同），还应该注意到聚类与主题具有完全不同的特征。 例如，群集22（“广告”）将其主要关于墙纸的应用程序分组，并在描述中提及该应用程序正在使用广告。 这与群集32（“设置和壁纸“），例如，它也将应用程序分组

是关于壁纸的，但在描述中不提及广告。

每个应用程序一个群集 就像现在一样，每个应用程序都属于一个集群，可能包含多个主题。 这导致了一个

良好的类似应用程序集群。 尚未探索的替代方案是允许应用程序成为多个群集的成员。 这个

可能会提供更好的聚类结果。

聚类方法的选择。 在使用K-means之前，我们尝试了正式的概念分析以检测相关的概念

主题和功能的细节 [25] 没有成功的结果; 该分析被应用程序和功能的数量所淹没。 K-means有已知的局限性，我们相信

其他聚类算法可以改善聚类。

低质量的应用。 像Google Play商店这样的应用商店包含

几个可疑值的免费应用程序。 将我们的方法限制为最低的下载次数或用户评级可能会产生非常不同的结果。 但是，目标

我们的方法是在用户看到它们之前识别异常值

因此我们应该考虑所有应用程序

3. 通过APIS识别出口者

现在我们已经根据其描述主题的相似性来聚类应用程序，我们可以搜索有关其实际行为的异常值。 第3.1节 展示了我们如何从中提取API功能

30.	主题壁纸	593 壁纸（90%），假期和
31.	“个性化”	402 宗教，分享 个性化（86%），共享，设置和使用情况
32.	“设置和壁纸”	251 **设置和使用率（37%），壁纸（37%），个性化**

伦敦餐厅应用程序最终以Cluster 12的形式结合其他主要关于导航和旅行的应用程序。 这些应用程序的集群通过它们的描述相关，我们现在可以根据它们的行为搜索异常值。

7. 备选集群方法

与大多数科学工作一样，本文中提出的方法只是经过了几次弯路，死路一条，

Android二进制文件。 第3.2节侧重于由权限控制的API。 第3.3节 描述CHABADA如何检测API异常值。

3.1 提取API使用情况

正如介绍中所讨论的那样，我们使用静态API作为a行为代理。 使用API的使用非常简单：Android字节码也可以进行高级静态分析

例如信息流分析和标准混淆技术，这些技术很容易阻碍任何静态分析，因此必须明确声明API的用法；在Android二进制文件中，与其他平台上的大多数二进制文件一样，静态API的使用很容易提取。 对于每个Android应用程序，我们使用apktool提取（二进制）APK文件[6]，并且使用小型反汇编程序，我们提取了所有API调用，包括每个API的调用站点数量。

[6] https://code.google.com/p/android-apktool

表4：伦敦餐厅使用的敏感API。 大胆
API使得这个应用程序在其集群中异常。

android.app.NotificationManager.notify（）
java.net.URL.openConnection（）
android.telephony.TelephonyManager.getDeviceId（）
android.app.NotificationManager.notify（）
android.app.NotificationManager.notify （）
org.apache.http.impl.client.DefaultHttpClient（）
org.apache.http.impl.client.DefaultHttpClient.execute（）
android.location.LocationManager.getBestProvider（）
android.telephony.TelephonyManager.getLine1Number（）android。
net.wifi.WifiManager.isWifiEnabled（）
android.accounts.AccountManager.getAccountsByType（）
android.net.wifi.WifiManager.getConnectionInfo（）
android.location.LocationManager.getLastKnownLocation（）
android.location.LocationManager.isProviderEnabled（）
android.location .LocationManager.requestLocationUpdates（）
android.net.NetworkInfo.isConnectedOrConnecting（）
android.net.ConnectivityManager.getAllNetworkInfo（）

2. 敏感的API

使用所有API调用作为功能将导致后期过度配置。 因此，我们选择了一部分API，即由Android权限设置管理的敏感API。 这些API访问敏感信息（例如用户的图片库，相机或麦克风）或执行敏感任务（更改系统设置，发送消息等）。安装应用程序时，用户必须明确允许使用这些API。 为此，每个Android应用程序都包含一个清单文件，其中列出了应用程序执行所需的权限。 为了获得一组敏感的API，我们依靠Felt等人的工作，他们确定并使用权限和Android方法之间的映射[7]; 我们认为只有在二进制文件中声明该应用程序并且在清单文件中请求其相应权限时，该应用程序才会使用该敏感API。这使我们能够消除在第三方库中使用的API调用，而不是直接由应用程序使用。

作为这种敏感的API的例子，请考虑 表4。 这些是由伦敦餐厅应用程序使用的API，将受特定权限的约束。 通过这些API，应用程序通过HTTP连接访问当前的网络提供者，最后一个已知位置（以及更新）和Internet。 这些API还显示应用程序访问设备标识符，用户的帐户信息以及手机“line1”号码。 后面这些以粗体显示的API将受“GET-ACCOUNTS”权限的约束。 由于每个权限都管理着多个API，单独使用权限会导致我们只能学习很少的功能。 相反，敏感的API允许对应用程序行为进行更细致的表征。

3. 使用OC-SVM识别API异常值

既然我们拥有所有应用程序的所有API功能，那么下一步就是识别异常值 - 即那些在相应主题群集中API使用情况异常的应用程序。 为了识别这些异常值，我们使用一类支持向量机器学习（OC-SVM） [22], 这是一种学习一类元素特征的机器学习技术。 由此产生的支持向量机模型可以用于此类中的异常/新颖性检测。 （请注意，这与支持向量机作为分类器的更常见用法形成对比，其中每个应用程序还必须被标记为属于特定的类 - 比如说，“良性”与“恶意” - 培训期间）。

OC-SVM已成功应用于从文档分类跨越的各种环境中 [17] 自动检测异常的Windows注册表访问 [10]. 在我们的上下文中，OC-SVM的有趣特征是人们只能提供一个类的样本（比如常规的良性应用），分类器将能够识别属于同一类的样本，将其他人标记为异常。 因此，OC-SVM主要用于那些存在一类元素样本（例如良性应用程序）的样本很多的情况，以及其他类别（例如恶意应用程序）样本不多的情况。

通过敏感的API作为二进制特性，CHABADA会在每个集群中使用应用程序的子集来训练OC-SVM，以模拟该集群中的应用程序通常使用哪些API。 然后使用生成的特定于集群的模型来识别异常值应用程序，即使用的API与同一群集内API的常见用法不同的应用程序。

为了表示元素距常见行为的距离，我们使用OC-SVM构建的超平面元素的实际距离。 这个距离越大，元素来自通常观察到的行为。 因此，通过将元素（即应用程序）与OC-SVM超平面的距离排序，我们可以确定哪些元素具有与通常观察到的差异最大的行为。

例如，再次考虑我们的伦敦餐厅应用程序。 在集群12中的应用上训练OC-SVM之后，它将伦敦餐厅归类为异常值。 原因是API以粗体显示 表 4 - 实际上，访问设备标识符，用户的帐户信息或他的手机号码对于“导航和旅行”群集中的应用程序而言并不常见。 在我们的评估中 第4节， 我们讨论使应用成为异常的趋势。

4. 替代方法

为了检测异常情况，我们确定了几种替代设置; 其中一些我们已经尝试过。 再次，我们在这里简要列出它们。

类和包名称作为抽象。 在使用Felt等人的映射进行敏感的API之前，我们考虑通过仅考虑类名或包名而不是整个方法签名来抽象API方法调用。 虽然这有助于减少功能的数量，但也导致相关信息的丢失。 对敏感方法（如返回用户电话号码的getLine1Number（））的调用与相对无害的方法（如返回当前数据连接的getNetworkType（）无法区分，因为它们都在TelephonyManager类中声明。

呼叫站点的数量。 对于每个API，我们都会考虑一个二进制值（即应用程序中是否存在至少一个该API的调用站点）。 我们可以将每个API的呼叫站点的规范化数量视为一项功能。 我们期望类似的结果，尽管我们只用二进制值进行了实验。

用于API的TF-IDF。 由于一些API通常用于所有群集，例如用于Internet访问的API，因此我们可能只考虑群集中最相关且最具代表性的方法，而不是所有方法。 术语频率逆文档频率-IDF） [13] 可以滤除一些非判别特征，从而为OC-SVM算法提供更大的支持。 尽管我们计划在不久的将来进行调查，但我们还没有尝试过这条路。

不敏感的API。 为了避免过度配合，限制功能的数量至关重要。 因此，我们专注于API

“敏感”，即受到Android权限的影响。 将此集合扩展到其他相关的API可能会产生更好的结果。

权限而不是API。 我们可以使用清单文件中的权限列表作为功能，而不是API。 然而，研究表明，将近30％的Android应用程序需要比实际使用更多的权限 [7, 2, 3, 23]. 我们选择了API，因为它们提供了对应用程序做什么和不做什么的更细致的视图。

单独避免API作为预测因子。 在针对已知恶意软件的描述和API训练分类器时，恶意软件集中使用的特定API（通常是发送文本消息）将主导说明。 首先通过描述进行聚类，然后进行分类，从而获得更好的结果。

4. 评估

为了评估我们技术的有效性，我们调查了以下主要研究问题：

RQ1我们的技术能否有效识别Android应用程序中的异常（即描述和行为之间的不匹配）？ 为此，我们手动检查了由我们的技术产生的顶级异常值，并将它们分类为隐蔽行为 （部分 4.1).

RQ2我们的技术可以用于识别恶意Android应用程序吗？ 为此，我们在我们的应用程序中包含了一组已知的恶意软件，并且我们运行了OC-SVM作为分类器 （部分 4.2).

1. 离群值检测

让我们从*RQ1*开始：我们的技术能否有效识别*Android*应用程序中的异常（即描述和行为之间的不匹配）？ 为此，我们在所有的*32*个群集上运行了*CHABADA*，如章节中所述 *3*. 经过K-fold验证，我们将整套22,521个应用分为10个子集，我们使用9个子集来训练模型，1个用于测试。 我们跑了这10次，每次考虑一个不同的子集进行测试。 在每轮中确定的异常值列表中，我们从每个群集的排名列表中确定了前5个异常值。 现在必须评估这160个异常值是否真的会出现可疑行为。

1.1. 手动评估

最后，只有一个人可以正确解释应用程序描述中的内容。因此，对于这160个应用程序，我们会手动检查其描述，即所使用的API列表，以手动检查代码。 我们会将每个应用程序分为三类：

恶意 - 该应用程序使用敏感的API显示未经改动（隐蔽）的行为，这违反了用户的利益。

可疑 - 该应用程序使用敏感的API显示未被广泛（隐蔽）的行为，但不一定会违背用户的兴趣。

良性 - 所有敏感行为都被恰当地描述。 这包括明确列出他们收集的敏感数据的应用程序，以及由于描述不足而放置在错误群集中的应用程序。

除非证实有罪，否则我们应用“无辜”原则：平均而言，每次此类评估对于良性应用程序需要2分钟，对于可疑或恶意应用程序则需要5分钟。

表5 总结我们的评估。 总体而言，我们明确指出42个异常值为恶意软件，占我们样本的26％。 鉴于这些应用程序源于应该保持质量的应用程序商店，这是一个令人担忧的结果。[7] 更令人担忧的是，有大量的异常值都是恶意的应用程序集群。 在群集14（“孩子”），16（“连接”），28（“广告壁纸”），30（“主题壁纸”）中，大多数异常值是恶意的。

所有这一切的好消息是，CHABADA报道的异常值得研究：前5个异常值中有39％需要应用商店经理或最终用户进行额外审查，他们可能也会为了保护他们而经营CHABADA。

由*CHABADA*生产的顶级异常值包含*26*％的恶意软件*;* 额外

1.2. 什么使异常？

在我们调查过程中，我们发现了一系列重复的趋势，这些趋势确定应用程序是否属于异常值。 这些趋势可以表征如下：

间谍软件广告框架。

我们发现的大部分“恶意”应用程序都是间谍软件。 我们在不同群集中识别出多个应用程序，这些应用程序获取敏感信息，例如用户的电话号码，设备ID，用户的当前位置以及用于不同帐户（例如Google和Facebook帐户）的电子邮件列表。 应用程序不会自己使用这些信息，但他们只会检索这些数据，因为它们包含广告的第三方库，以及广告公司（例如apploving和airpush为应用程序开发人员支付用户的敏感信息）。 一些应用程序清楚地表明它们包含这样的框架，并且其中大多数结束于广告群集，在这种情况下这种行为是正常的。 然而，没有提及此类框架的使用及其对隐私的影响的应用程序是间谍软件。

仅举几个例子，我们发现了“蚊子杀手”应用程序，例如驱蚊剂 - 无广告，防蚊虫，驱蚊加，婚礼想法画廊等婚礼应用程序以及圣诞女孩壁纸集合等应用程序动态壁纸和暮光之城动态壁纸。 由于它们都包含相同的广告框架，因此它们都会获得上述敏感数据并将其发送到广告服务器。

可疑的行为。

在“可疑”应用程序中，其描述不能完全证明该行为。 例如，UNO Free游戏应用程序无需解释即可访问用户的位置，百老汇节目的官方应用程序WICKED可以录制音频，但并未说明用于哪些目的。 Runtastic是一个广泛使用的培训应用程序，它也可以录制音频，但在说明中没有提及它。 最后，Yahoo！ Mail是Yahoo浏览和撰写电子邮件的官方应用程序，可以发送短信。 从描述中不清楚为什么应用程序应该这样做。

错误分类的应用程序。

一些“良性”应用程序根据其描述被错误分类，因此被分配到集群中，

[7] 但是请注意，在下载时间和写作时间之间，许多这些应用程序已被用户识别为恶意软件，并且已被删除。

表5：手动评估前5个异常值，每个群集和总数。

行为	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	总
恶毒						1	2	1	0	0	0	0	2	0	0	0	3	0	4	2 (26%)	0	0	1	3	5	1	3	1	1	2	3	1	4	1	1	**42**
可疑						1	2	1	0	0	0	0	0	0	2	2	1	1	0	0	1 (13%)	2	1	1	0	0	2	0	1	0	1	0	1	0	0	0	**20**
良性				3	1	3	5	5	5	5	3	5	3	3	4	4	2	5	1	2 (61%)	3	4	3	2	0	2	2	3	4	2	2	3	1	4	4	**98**

由涵盖完全不同主题的应用程序组成。 例如，SlideIT免费键盘是一种通过沿键盘字母滑动手指插入文本的流行工具，最终成为语言集群，因为超过一半的描述是关于语言支持的。 罗马尼亚赛车是一款赛车游戏，它最终也进入了同一个集群，主要原因是它的简短描述提到了多语言支持。

我们还有一些罕见的应用程序与完全意外的群集相关联。 仓鼠生活，一个宠物训练游戏，结束了“宗教壁纸”群集。

罕见的行为。

有些应用程序被标记为异常值，因为它们的行为虽然是良性和清晰的描述，但对于这些应用程序所属的群集而言很少见。 例如，SoundCloud - Music & Audio被标记为异常，主要是因为它记录了音频。 这是预期的行为，因为应用程序连接到查找和共享新音乐的平台，并且允许录制用户制作的音乐。 但是，录制音频不是Cluster 1的常见功能之一，因此SoundCloud被标记为异常。 同样，Llama - 允许根据上下文和位置更改铃声的位置配置文件被标记为异常，因为个性化应用访问用户的位置和日历并不常见。 不过，该应用程序使用此信息在用户处于家中，办公室或开会时自动在振动和铃声之间切换。

良好的异常值。

在一些集群中，CHABADA将不常见的行为确定为缺乏恶意行为。 例如，Cluster 13（“money”）包含多个使用库进行广告的应用程序，从而访问敏感信息。 因此，威尔先生的梭哈扑克和威尔先生的抽奖扑克被标记为异常，因为他们不访问敏感信息，尽管他们是扑克游戏。

*CHABADA*是行为不可知的：它不能确定应用程序是好还是

2. 恶意软件检测

现在让我们转向RQ2：我们的技术可以用于识别恶意Android应用程序吗？ 为此，我们使用了Zhou等人的数据集。[28] 包含超过1,200个针对Android的已知恶意应用程序。 这些应用程序的原始形式缺少标题或描述等元数据。 由于这些应用中有很多都是原始应用的重新包装版本，因此我们能够从Google Play商店收集适当的说明。 我们使用应用程序的标题和包标识符来搜索商店中的正确匹配。 对于72个案例，我们可以找到完全相同的包标识符，而对于116个应用程序，我们发现包标识符非常相似的应用程序。 我们手动检查了比赛是否正确。 就像我们最初的一套“良性”应用程序一样 （部分2.1），我们只保留这些应用程序中的英文说明，将其减少到172个应用程序。

作为恶意软件检测器，我们再次使用OC-SVM模型；但是这一次，我们将它用作分类器 - 也就是说，我们使用SVM模型对一个元素是否是同一分布的一部分进行了二元决策。

2.1. 使用主题群集分类

我们在前面的研究中使用的“良性”应用程序集上运行了分类，我们包含了172个恶意软件样本，我们可以找到相应的描述。 我们只像以前一样在“良性”应用程序上训练OC-SVM，并且我们排除了在之前的实验中手动分类为“恶意”的应用程序 （部分 4.1）；然后，我们在每个群集内对这些应用程序中90％的API进行OC-SVM训练，然后使用OC-SVM作为由该群集中已知恶意应用程序组成的测试集的分类器，以及剩余的10％良性的应用程序。 我们因此模拟的是恶意软件攻击完全新颖的情况 - CHABADA必须在不知道以前的恶意软件模式的情况下正确识别恶意软件。

至于之前的实验，我们会这样做10次，每次都考虑不同的测试集。 恶意应用程序的数量不会在集群中平均分配，因为根据恶意应用程序的描述将恶意应用程序分配给集群。 在我们的评估设置中，使用我们的数据集，每个群集的恶意应用程序数量从0到39。

我们的分类结果显示在 表6。 我们报告10次不同运行的平均结果。 CHABADA正确识别56％的恶意应用程序，而只有16％的“良性”应用程序被错误分类。 如果我们的方法将用于防范未知的恶意行为，它会检测到大部分恶意软件。

表6：检查主题集群中的API和描述（我们的方法）

	预测为恶意	预测为良性
恶意应用程序	96.5 (56%)	75.5 (44%)
良性的应用程序	353.9 (16%)	1,884.4 (84%)

与标准的恶意软件检测器相比，这些结果当然有待改进 - 但这是因为现有的恶意软件检测器与已知的恶意软件进行比较，恶意软件的签名和行为已知。 例如，像伦敦餐厅应用程序那样访问设备上的所有用户帐户是已知的恶意行为模式。 实际上，我们的方法将用于补充这种检测器，并专门针对与现有恶意软件不同的新型攻击，但其API使用情况足够异常并被标记为异常值。

2.2. 没有聚类的分类

我们通过与替代方法进行比较来进一步评估我们方法的有效性。 为了显示主题聚类的影响，我们将我们的分类结果与其中的设置进行比较

在我们的示例中，即使不知道现有的恶意软件模式，

OC-SVM将从敏感的API和单独的描述中对NLP预处理后的单词进行培训 - 也就是说，所有应用程序形成一个大集群。 如表7 显示，恶意软件检测率显着下降。 这显示了我们的聚类方法的好处。

表7：在一个群集中检查API和说明

	预测为恶意	预测为良性
恶意应用程序	41 (24%)	131 (76%)
良性的应用程序	334.9 (15%)	1,903.1 (85%)

没有集群的分类会产生更多的假阴性。

2.3. 使用给定的类别分类

最后，人们可能会问，为什么我们需要基于描述主题的特定聚类方法，因为人们也可以很容易地使用给定的商店类别。 为此，我们根据Google Play商店中的类别对应用程序进行了集中，并重复了实验以及最终的30个集群。 结果 （表8）证明了聚类的一般好处； 然而，像我们的方法那样，话题聚类仍然明显优越。 （另外，有人可能会认为某个类别是某些图书管理员会分配的东西，因此需要更多的工作和更多的数据。）

表8：检查Google Play商店类别中的API和说明

	预测为恶意	预测为良性
恶意应用程序	81.6 (47%)	90.4 (53%)
良性的应用程序	356.9 (16%)	1,881.1 (84%)

通过描述主题进行聚类优于给定类别的聚类。

4.3 有效性的限制和威胁

像任何实证研究一样，我们的评估受到有效性威胁，其中许多是由我们的方法局限性引起的。 下面列出了最重要的威胁和限制。

外部有效性。 CHABADA依靠现有的，假设的大多数是良性的应用程序来建立描述主题和程序特征之间的关系。 我们不能声称所述关系可能适用于其他应用程序生态系统，或者可以转移到其他应用程序生态系统。 我们已经记录了我们的步骤，以便轻松复制我们的方法。

仅限免费应用。 我们的22,521个应用程序样本仅基于免费应用程序； 即需要通过广告，购买或捐赠产生收入的应用程序。 不考虑付费应用程序使我们的数据集有偏见。 然而，这种偏见会使“正常”更多地转向广告和其他收入方式支持的应用，这些方式更接近恶意软件暴露的不良行为。 因此，我们的结果是保守的，宁可通过更大比例的付费应用程序来改善，这可以预期是良性的。

应用程序和恶意软件偏见。 我们的示例仅反映了Google Play商店中每个类别的前150次下载。 此示例偏向于经常使用的应用程序，并偏向于较少使用的类别； 同样，我们选择恶意软件 （部分 4）可能或可能不代表当前的威胁。 不知道Android用户使用哪些实际的应用程序以及如何使用这些应用程序，这些示例可能有偏差。 再次，我们允许轻松复制我们的方法。

研究人员的偏见。 我们对异常值的评估是基于一个人的分类，他是本文的共同作者。 这带来了研究者偏见的风险，即作者想要获得最佳结果的愿望。 为了应对这种威胁，我们正在公开我们的数据集（仲 - 重刑 6).

原生代码和混淆。 我们将分析限制在Dalvik字节码。 我们不分析本地代码。 因此，应用程序可能依赖本地代码或使用混淆来执行隐藏行为; 但随后，这些特征可能再次表征异常值; 而且，这些都不会改变必须调用的API集合。

静态分析。 由于我们依赖静态API使用，因此我们受到静态分析典型的限制。 特别是，我们可能会错过通过反射引发的行为，即在运行时生成的代码。 尽管存在使用反射静态分析Java代码的技术，但这些技术并不直接适用于Android应用程序 [5，8]; 从长远来看，与测试生成配对的动态分析可能是更好的选择。

静态API声明。 由于我们会静态提取API调用，因此我们可能会考虑应用程序永远不会执行的API调用。 静态检查API是否到达是（不可判定的）暂停问题的一个实例。 作为一种解决方法，我们决定仅在清单中声明相应权限的情况下才考虑使用API。

敏感的API。 我们检测敏感的API （第3.2节） 依靠Felt等人的绘图。 [7]，现在，两年后，可能会部分过时。 映射中的条目不正确或缺失会导致CHABADA错过或错误分类应用程序的相关行为。

5. 相关工作

虽然这项工作可能是第一个根据应用程序行为来检查应用程序描述的第一个工具，但它建立在结合自然语言处理和软件开发的先前工作的历史上。

1. 挖掘应用程序描述

与我们的工作最相关的是Pandita等人的WHYPER框架。 [19]. 就像我们的方法一样，WHYPER试图自动化Android应用程序的风险评估，并将自然语言处理应用于应用程序描述。 WHYPER的目标是确定是否需要敏感的权限（如访问联系人或日历）是在应用程序描述中激发的。 与CHABADA完全自动学习哪些主题与哪些API相关联（以及扩展，哪些权限）相比，WHYPER需要手动注释描述需要权限的句子。 此外，CHABADA在两个方面超越了权限：首先，它专注于提供更详细视图的API，它旨在解决期望和实现之间的一般不匹配问题。

哈曼等人在一年前就提出了应用商店挖掘的想法。 挖掘黑莓应用商店 [9]. 他们专注于应用程序元数据以查找相关消费者评级和应用程序下载排名等模式，但不会自行下载或分析应用程序。

我们对“正常”行为的描述来自采矿相关应用; 一般来说，我们假设维护良好的商店中的大多数应用程序也是大多数用户期望合法的。 相反，Lin等人最近的工作。 [14] 提示

众包来推断用户对特定隐私设置的期望； 就像我们发现的那样，Lin等人 还强调隐私预期因应用类别而异。 用户提供的这些信息可以很好地补充我们从应用描述中推断出的内容

2. 行为/描述不匹配

我们的方法也与使用自然语言处理从评论和文档推断规范的技术相关。 林坦等人。 [24] 从程序语料库中提取隐式程序规则，并使用这些规则自动检测注释和源代码之间的不一致性，从而指示错误或错误评论。 规则适用于调用和资源访问的排序和嵌套（不得从f_b“调用”f_a“）。

Høst和Østvold [11] 向程序语料库学习哪些动词和短语通常与特定的方法调用相关联，并用它们来识别错误的方法。

Pandita等人 [20] 从超过2,500个API文档句子中识别描述代码合同的句子; 这些合同可以通过测试或静态分析进行检查。

所有这些方法都将程序代码与正式程序文档进行比较，这些程序代码的半正式特性使得更容易提取需求。 相比之下，CHABADA的工作是最终用户文档，与程序结构分离。

3. 检测恶意应用程序

有大量的工业产品和研究原型专注于识别已知的恶意行为。 对我们的工作影响最大的是周和江的论文 [28], 他们使用应用程序请求的权限作为过滤器来识别潜在的恶意应用程序; 实际检测使用静态分析来比较API调用序列与已知恶意软件的调用序列。 与所有这些方法相比，CHABADA即使不知道是什么导致恶意行为，也能识别出异常值。

TAINTDROID系统 [6] 跟踪Android应用程序中的动态信息流，从而可以检测敏感信息的使用情况。 使用这样的动态流信息将产生比静态API使用更精确的行为见解； 同样，Profilerroid等分析器 [26] 会提供更好的信息； 但是，TAINTDROID和ProfileDroid都需要一组具有代表性的执行程序。 将这些技术整合到CHABADA中，并结合自动化测试生成 [12, 27, 15, 1], 将允许学习信息流的正常和异常模式; 这是我们未来工作的一部分 （部分 6).

6. 结论和后果

通过按描述主题对应用进行聚类，并根据每个集群内的API使用情况识别出异常值，我们的CHABADA方法可以有效地识别在其描述中行为会出乎意料的应用程序。 我们发现了几个虚假和误导性广告的例子。 并且作为副作用，获得了用于未知恶意软件的新型有效检测器。 就像采矿软件档案为经验软件工程开辟了新的机遇，我们看到挖掘应用程序及其描述为自动检查自然语言需求开辟了一些新的机会。

在我们的工作中，我们获得了对需要采取行动的Android应用程序生态系统的许多见解。 首先，应用程序供应商必须更清楚地了解他们的应用程序如何赚取收入。 谷歌等应用商店供应商应该引入更好的标准，以避免欺骗或不完整的广告。 其次，Android向用户提供权限的方式被打破。 普通用户不会理解“允许访问设备标识符”的含义，也没有办法检查他们的敏感数据实际上正在做什么，他们也不会理解后果。 不过，用户了解常规应用的功能， CHABADA将指出并突出差异，这应该更容易掌握。

虽然我们现在的方法是通过探索和改进几种替代方法，但我们清楚地意识到它并不完美或完整。 我们未来的工作将重点关注以下主题:

详细的行为模式。 静态API的使用是一个相当广泛的抽象，用于表征应用程序的功能和不具有的功能。 更高级的方法可以专注于API的交互，特别是API之间的信息流。

动态行为。 探索实际执行情况可以更详细地了解应用程序实际执行的操作，特别是访问远程资源的所有API的具体值。 我们正在研究Android应用程序的GUI测试生成器，旨在覆盖特定的API或动态信息流。

自然语言处理。 自然语言处理领域的最新技术可以检索的不仅仅是主题。 查看单词之间的依赖关系（例如连词，主语动词，动词宾语）可以检索更详细的模式。 同样，利用已知的本体将有助于识别同义词。

主题和行为的罗塞塔石。 通过挖掘数千个应用程序，我们可以将自然语言描述与特定程序行为相关联。 由此产生的自然语言和程序片段之间的映射有助于程序理解以及程序和测试的综合。

为了便于复制和验证我们的工作，我们打包了本工作中使用的所有数据以供下载。 具体而言，我们准备了一个50 MB数据集，其中包含应用名称，描述，其他元数据，权限和API使用情况等进入CHABADA的确切数据。 所有这些都可以在CHABADA网站上找到:

`http://www.st.cs.uni-saarland.de/chabada/`

7. 致谢

我们感谢Vitalii Avdiienko，Juan Pablo Galeotti，Clemens Hammacher，Konrad Jamrozik和Sascha对本文早期版本的有益反馈。特别感谢Andrea Fischer，Joerg Schad，Stefan Richter和Stefan Schuh在项目期间给予的宝贵帮助。

这项工作是由欧洲研究委员会（ERC）高级资助“SPECMATE - 规范挖掘和测试”资助的。

8. 参考

1. D. Amalfitano，AR Fasolino，P. Tramontana，S. De Carmine和AM Memon。 使用GUI抓取来自动测试Android应用程序。 在IEEE / ACM国际自动化软件工程会议（ASE），第258-261页，纽约，纽约，美国，2012年。ACM。
2. KWY Au，YF Zhou，Z. Huang和D. Lie。 PScout：分析Android权限规范。 ACM计算机和通信安全会议（CCS），第217-228页，纽约，纽约，美国，2012年。ACM。

3. A. Bartel，J. Klein，M. Monperrus和Y. Le Traon。 通过减少攻击面自动保护基于权限的软件：Android应用程序。 在IEEE / ACM国际自动软件工程会议（ASE），第274-277页，2012年。
4. DM Blei，AY Ng和MI Jordan。 潜在Dirichlet分配。 机器学习研究杂志，3：993-1022,2003。
5. E. Bodden，A. Sewe，J. Sinschek，H. Oueslati和M. Mezini。 驯服反射：在存在反射和自定义类加载器的情况下帮助进行静态分析。 在ACM / IEEE国际软件工程会议（ICSE），第241-250页，纽约，纽约，美国，2011年。ACM。
6. W. Enck，P. Gilbert，B.-G. Chun，LP Cox，J. Jung，P. McDaniel和AN Sheth。 TaintDroid：用于智能手机实时隐私监控的信息流跟踪系统。 在USENIX会议上
操作系统设计与实现（OSDI），1-6页，美国加州伯克利，2010年。USENIX协会。
7. AP Felt，E. Chin，S. Hanna，D. Song和D. Wagner。 Android权限被揭秘。 ACM计算机和通信安全会议（CCS），第627-638页，纽约，纽约州，美国，2011年。ACM。
8. C.Fritz，S.Arzt，S.Rasthofer，E.Bodden，A.Bartel，J.Klein，
Y. le Traon，D. Octeau和P. McDaniel。 对Android应用程序进行高度精确的污点分析。 技术报告TUD-CS-2013-0113，EC SPRIDE，2013。
9. M. Harman，Y. Jia和Y. Zhang。 应用商店挖掘和分析：应用商店的MSR。 在IEEE采矿软件仓库（MSR）工作会议上，第108-111页，2012。
10. KA Heller，KM Svore，AD Keromytis和SJ Stolfo。 一类支持向量机用于检测异常Windows注册表访问。 在ICDM计算机安全数据挖掘研讨会（DMSEC）上，2003。
11. EWHøst和BMØstvold。 调试方法名称。 在欧洲面向对象编程会议（ECOOP），第294-317页。 施普林格，2009年。
12. C.胡和我Neamtiu。 自动化Android应用程序的GUI测试。 在软件测试自动化国际研讨会（AST），第77-83页，纽约，纽约，美国，2011年。ACM。
13. KS琼斯。 术语特异性的统计解释及其在检索中的应用。 Journal of Documentation，28（1）：11-21,1972。
14. J. Lin，S. Amini，JI Hong，N. Sadeh，J. Lindqvist，and J. Zhang。 期望和目的：通过众包理解用户对移动应用隐私的心理模型。 在ACM无处不在计算会议（UbiComp），第501-510页，纽约，纽约，美国，2012年。ACM。
15. A. *Machiry*，*R. Tahiliani*和*M. Naik*。 *Dynodroid*：*Android*应用程序的输入生成系统。 在与*ACM SIGSOFT*软件基础国际研讨会共同举办的欧洲软件工程大会上*E*ngineering（ESEC / FSE），第224-234页，纽约，纽约州，美国，2013年。ACM。
16. JB MacQueen。 一些分类和分析多元观测的方法。 在LML Cam和J. Neyman编着的伯克利数学统计和概率专题讨论会上，第1卷，第281-297页。 加州大学出版社，1967年。
17. LM Manevitz和M. Yousef。 用于文档分类的一类SVM。 Journal of Machine Learning Research，2：139-154，2002。
18. AK McCallum。 Mallet：用于语言工具包的机器学习。 http://mallet.cs.umass.edu， 2002.
19. R. Pandita，X. Xiao，W. Yang，W. Enck和T. Xie。 WHYPER：实现移动应用程序自动化风险评估。 在USENIX安全研讨会，第527-542页，2013年。
20. R. Pandita，X. Xiao，H. Zhong，T. Xie，S. Oney，and A. Paradkar。 从自然语言API描述中推断方法规范。 在ACM / IEEE国际软件工程会议（ICSE）上，2012。
21. P. Rousseeuw。 剪影：图形辅助解释和验证聚类分析。 Journal of Computational and Applied Mathematics，20（1）：53-65,1987。
22. B.Schölkopf，JC Platt，JC Shawe-Taylor，AJ Smola和RC Williamson。 估计a的支持高维分布。 Neural Computation，13（7）：1443-1471,2001。
23. R. Stevens，J. Ganz，P. Devanbu，H. Chen和V. Filkov。 询问（和关于）Android应用使用的权限。 IEEE采矿软件仓库工作会议（MSR），第31-40页，加利福尼亚州旧金山，2013年。
24. L. Tan，D. Yuan，G. Krishna和Y. Zhou。 / * iComment：错误或不好的评论？ * /。 在ACM SIGOPS操作系统原理研讨会（SOSP），第145-158页，2007年。
25. A. *Wasylkowski*，*A. Zeller*和*C. Lindig*。 检测对象使用异常。 在与*ACM SIGSOFT*软件工程基础国际研讨会（*ESEC / FSE*）共同举办的欧洲软件工程会议上，第*35-44*页，纽约，纽约，*2007*年。*ACM*。
26. X. Wei，L. Gomez，I. Neamtiu和M. Faloutsos。 ProfileDroid：Android应用程序的多层剖析。 在ACM年度移动计算和网络国际会议（MobiCom），第137-148页，纽约，纽约，美国，2012年。ACM。
27. W. Yang，MR Prasad和T. Xie。 用于自动GUI模型生成移动应用程序的灰色方法。 在软件工程基础方法国际会议（FASE），第250-265页，柏林，海德堡，2013年。Springer-Verlag。
28. Y. Zhou和X. Jiang。 剖析Android恶意软件：特征和演变。 在IEEE Symposium on Security and Privacy（SP），第95-109页，华盛顿特区，美国，2012年。IEEE计算机协会。